A NEURAL NETWORK APPROACH TO CLASSIFYING GENERIC EXPRESSIONS

An Undergraduate Research Scholars Thesis

by

CARLO JACOB DE GUZMAN

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Ruihong Huang

May 2017

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

A Neural Network Approach to Classifying Generic Expressions

Carlo Jacob De Guzman
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University

Generic expressions make statements about nonspecific entities. Characterizing generic expressions make broad statements about classes of entities, without belonging to any temporal structure, while habitual generic expressions refer to regularly occurring events and actions. Non-generic expressions can also be classified as characterizing or habitual, complicating the task of differentiating between generic and non-generic expressions. Correctly differentiating generic expressions from non-generic expressions plays a role in tasks such as information extraction and knowledge base population, as this distinction determines the type of information that can be interpreted from a given expression.

The goal of this research is to develop a neural network that classifies generic expressions. Previous machine learning approaches to classifying generic expressions required precise feature extraction prior to the training process, and did not fully utilize semantic information in learning to classify generic expressions. A neural network approach will allow the system to learn from a wide variety of grammatical and semantic features, and adjust its internal model to take advantage of features that strongly identify an expression as generic or non-generic.

ACKNOWLEDGEMENTS

TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.

CHAPTER I

INTRODUCTION

Generic Expressions

Making the distinction between generic and non-generic expressions determines how the information in a given expression should be interpreted, as a generic expression contains information that applies to a wide variety of entities, and can be used to develop a more natural understanding of a general entity, while a non-generic expression contains information about a specific entity or limited group of entities, and can be used to answer questions with specific answers (Friedrich and Pinkal, 2015). When distinguishing generic expressions from other types of expressions, various types of information can be taken into account. For example, the identification of the subject as generic or non-generic can be used to determine whether the expression as a whole is generic or non-generic, as seen in the sentences below:

- **Humans** live for about seventy years. (generic)
- **Michael Smith** only lived to the age of 43. (non-generic)

However, the subject alone cannot adequately determine the genericity of an expression, as a generic entity can be used as a placeholder to describe the specific but unknown subject of a non-generic clause; this phenomenon appears in the sentences below:

- **Crocodiles** belong to the same order as alligators (generic)
- **Crocodiles** emerging from the lake scared the tourists (non-generic)

Thus, the classification of generic expressions requires information from both the subject and clause of a given phrase, as using only clause information would result in the incorrect classification of an expression involving a non-generic subject performing a generic action. Additional pieces of information, such as the usage of temporal structures and grammatical

features, also play a role in determining the genericity of a given expression. The vast amount of information available in a single expression makes the process of determining the features that play the most significant role in determining genericity relatively difficult.

Neural Networks

Neural networks are characterized by their usage of units known as *artificial neurons* to facilitate the learning process. Each individual neuron takes in all of the output data from the previous layer of neurons, performs a weighted function (usually a sum) to process the data, then sends a modified version of the result to the next layer of neurons, using a non-linear function to scale the result of its computation (Goldberg 2015).

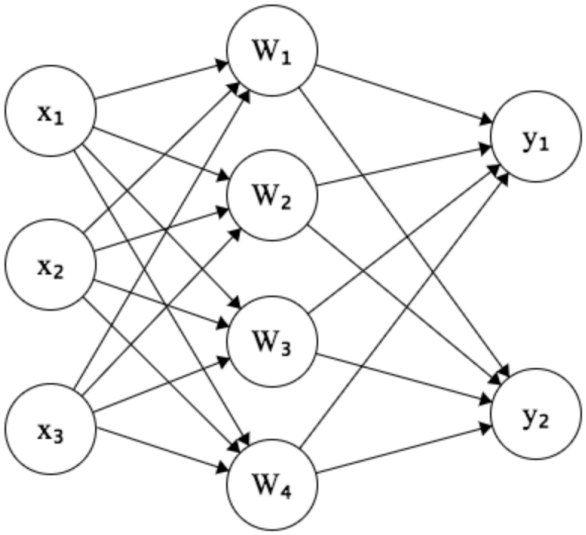

Figure 1: A simple feed-forward neural network

The model in Figure 1 is known as a *feed-forward neural network*, and can be represented in mathematical terms by equation (1):

$$\mathrm{NN}(\boldsymbol{x}) = g_2(g_1(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2) \quad (1)$$

In equation (1), **x** is the input vector (in this case, a set of numbers which represents a natural language expression), $\mathbf{W_i}$ is a weight matrix representing a given layer of "neurons", and *g(n)* is a non-linear function called an *activation function* that allows the neural network to model non-linear data. Common functions used for *g(n)* include the sigmoid function and the hyperbolic tangent function. Each layer can additionally be modified by adding a bias vector **b** if needed to better model the desired outcome.

Training a neural network involves minimizing a predefined *loss function*, which calculates the difference between the desired output and the actual output. Based on the calculations of the loss function the model parameters (i.e. the weight matrices and any applicable bias vectors) adjust their values to better fit the data previously seen. One such algorithm that performs this process is known as *stochastic gradient descent*, and involves calculating the gradients of the loss function with respect to the model parameters using *backpropagation*, then modifying the model parameters based on those gradients, and repeating the process to further minimize the loss.

The types of activation functions used in the neural network can affect gradient behavior in the learning process. For activation functions where the outputs lie in a fixed range (such as the aforementioned hyperbolic tangent), the gradients can drop close to zero and seemingly "vanish", while unbounded activation functions can face the opposite problem and have "exploding" gradients that grow too large. Though exploding gradients can be avoided by capping the gradient at a maximum value, vanishing gradients lack a surefire solution, and can only be mitigated by modifying the model to reduce the impact of shrinking gradients.

CHAPTER II

METHODS

WikiGenerics

Friedrich et al. (2015) developed the WikiGenerics corpus to provide an annotated data set for training automatic systems to classify generic expressions. This corpus consists of a collection of 104 Wikipedia articles each divided into individual phrases consisting of a subject and a clause. Three different annotators identified the subject and clause of each phrase as either generic or non-generic, with these two classifications resulting in three possible labels for a given phrase: GEN_gen (where both the subject and the clause are generic), NON-GEN_non-gen (where both the subject and the clause are non-generic), and NON-GEN_gen (where a generic subject has a non-generic clause). The authors note that this classification system does not allow for phrases with a non-generic subject and a generic clause (i.e. GEN_non-gen), as a generic clause makes characterizing statements about types of entities, so that it cannot unique characterize a non-generic subject. The gold standard labels then follow from the labels agreed upon by the majority of the annotators.

When looking at the labels assigned to each phrase, only the gold standard labels are considered for classification. To simplify the classification process, the combined subject-clause labels are condensed into "GENERIC" and "NON-GENERIC". If a given phrase had both its subject and clause classified as "GENERIC" by the gold standard labels, then it is considered "GENERIC"; otherwise, the phrase is considered "NON-GENERIC". Additionally, I also extract grammatical information from each phrase using the part-of-speech tagger included in Python's NLTK

(Natural Language Toolkit) library. Both the words in a given phrase and their associated part of speech tags will be considered in the classification process, but they first need to be converted to a numerical representation that the neural network can use to tune its internal model.

Word2vec

Developed by Tomas Mikolov and other researchers at Google, the word2vec model takes in a collection of documents as its input and develops vector representations of the vocabulary in all the documents by considering the contexts in which the words are used. As opposed to the traditional one-hot representation, in which distinct words have distinct representations as sparse vectors of ones and zeros, the representation of words as dense vectors allows for similar words to have similar vectors, and can demonstrate various relationships between different words (Goldberg 2015). These word vectors can be used in equations such as "king" – "man" + "woman" = "queen"; the previous equation shows that the model recognizes that "queen" can be defined as the female version of a king, and their corresponding word vectors show that various semantic components can be added and removed to form the word vectors for related words.

This model uses word vectors trained by Mikolov et al. (2013) on the Google News dataset consisting of around 100 billion words. For each word in a given phrases, its corresponding word vector is extracted and added to an aggregate of the previous vectors. Because the clauses vary in length, a *continuous bag of words* representation of the phrase is used to ensure that the neural network receives inputs of the same size. This representation involves adding all of the word vectors together, then averaging them based on the number of words in the phrase so that the length of a phrase does not affect its classification. Additionally, a word2vec model for the part-

of-speech tags was trained using gensim, a topic modeling library developed by Radek Rehurek. In a similar fashion to the word vectors, the part-of-speech vectors are added and averaged into a single vector, which is then appended to the averaged word vector to create a vector consisting of word data and part-of-speech data that will serve as input to the neural network. Because the word vectors were not trained on the WikiGenerics corpus, they do not contain representations for every single word. As a consequence, some of the vectors do not fully represent their corresponding phrase, as they do not contain any information for words that lack a vector representation.

Neural Network Model

The TensorFlow library allows for the representation of machine learning networks as computational graphs, which do not perform any actual computations until run. The model developed in TensorFlow is a *feed-forward* neural network that takes in vectors representing word embeddings for a given phrase and its grammatical features, and outputs a classification of the given phrase as either generic or non-generic. This model consists of an input layer, a hidden layer that transforms the input based on self-determined weights, and an output layer which produces a result based on calculations in the hidden layer; these layers are all *fully connected*, meaning that each unit contained in a given layer receives all outputs from the previous layer as its input, then sends its computations to all other units in the next layer.

The hidden layer of the neural network uses hyperbolic tangent as its activation function, while the output layer uses the softmax function as its activation function. The softmax function, seen

below in equation (2), converts the values in the output layer to probabilities representing the likelihood of a given phrase having a certain classification.

$$softmax(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}}}{\sum_i e^{x_i}} \quad (2)$$

For this model, the network will use the classification with a higher probability as its prediction. In the training process, the network then computes the cross-entropy between the predicted classification and the classification assigned in the training data, and uses this value as the loss to minimize. During the training phase, the neural network will update its weights through backpropagation after each input phrase receives a predicted classification.

CHAPTER III

RESULTS

In order to limit the effects of overfitting during training, the WikiGenerics data set was divided into a training set and a testing set. The training set consists of 9,154 phrases from 93 different articles, and the testing set consists of 1,201 phrases from 10 different articles. The hidden layer weights were initialized as random values following a normal distribution. Precision, recall, and F-scores were calculated for the model after 10 epochs and after 100 epochs, where an epoch refers to the neural network going through all of the training phrases once, in order to quantify the accuracy of the neural network. Table 1 shows the averages of the measured values for accuracy, precision, recall and F1-score over ten different training instances with the given number of epochs to account for differences in the initial random weights.

Table 1: Neural Network Measures of Accuracy

Data Set	Number of Epochs	Accuracy	Classification	Precision	Recall	F1
Training	10	0.76	Generic	0.755	0.753	0.753
			Non-Generic	0.764	0.769	0.765
	100	0.857	Generic	0.888	0.83	0.858
			Non-Generic	0.828	0.888	0.856
Testing	10	0.738	Generic	0.753	0.802	0.776
			Non-Generic	0.715	0.657	0.684
	100	0.762	Generic	0.798	0.808	0.802
			Non-Generic	0.708	0.697	0.701

CHAPTER IV

CONCLUSION

This paper presents the application of a feed-forward neural network to the task of classifying generic expressions, using both word embeddings and part-of-speech embeddings to represent semantic and grammatical features to learn from. Because this model used pre-trained word embeddings, it ignored words for which an embedding did not exist; finding quality embeddings for these missing words would improve the quality of the information that the neural network learns from. While the network can be trained to achieve similar levels of accuracy in classifying generic and non-generic expressions, the accuracy of classifying non-generic expressions falls behind the accuracy of classifying generic expressions when running the trained model on previously unseen data. Thus, future research would look into identifying features that distinctly identify phrases as non-generic, along with neural network models that can take advantage of these features.

REFERENCES

[1] Annemarie Friedrich, Alexis Palmer, Melissa Peate Srensen, and Manfred Pinkal. 2015. *Annotating genericity: a survey, a scheme, and a corpus*. In Proceedings of the 9th Linguistic Annotation Workshop (LAW IX), Denver, Colorado, US. Retrieved from http://www.aclweb.org/anthology/W15-1603.pdf

[2] Annemarie Friedrich and Manfred Pinkal. 2015. *Discourse-sensitive Automatic Identification of Generic Expressions*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Lingiustics and the 7th International Joint Conference on Natural Language Processing, Beijing, China. Retrieved from http://www.aclweb.org/anthology/P15-1123

[3] Yoav Goldberg. 2015. *A Primer on Neural Network Models for Natural Language Processing*. Retrieved from http://www.cs.biu.ac.il/~yogo/nnlp.pdf

[4] Steven Bird, Edward Loper and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc. http://www.nltk.org/

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. *Efficient Estimation of Word Representations in Vector Space*. https://code.google.com/archive/p/word2vec/

[6] Radim Rehurek and Petr Sojka. 2010. *Software Framework for Topic Modelling with Large Corpora*. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta. https://radimrehurek.com/gensim/index.html

[7] Dong Jin, Yu. 2016. *Part-Of-Speech Tag Embedding for Modeling Sentences and Documents*. UCLA Electronic Theses and Dissertations. Retrieved from http://escholarship.org/uc/item/0vk28220

[8] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. *TensorFlow: A System for Large-Scale Machine Learning*. 12th USENIX Symposium on Operating Systems Design and Implementation, Georgia. https://www.tensorflow.org/